#7

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 376 003 A2**

(12) # EUROPEAN PATENT APPLICATION

(21) Application number: 89122414.9

(22) Date of filing: 05.12.89

(51) Int. Cl.5 G06F 15/16

(30) Priority: 29.12.88 US 291890

(43) Date of publication of application:
04.07.90 Bulletin 90/27

(84) Designated Contracting States:
DE FR GB

(71) Applicant: International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: Bolan, Timothy Vincent
2922 Ricky Dr.
Endwell, NY 13760(US)
Inventor: Boston, Josephine Ann
206 Prospect Street
Endicott, NY 13760(US)
Inventor: Hanrahan, Donald Joseph
931 Byford Boulevard
Endwell, NY 13760(US)
Inventor: Laubli, Bernhard
300 Princeton St.
Endicott, NY 13760(US)
Inventor: Ring, David Arthur
1036 Schuyler St.
Endicott, NY 13760(US)
Inventor: Rundle, Alfred Thomas
2926 Twilight Dr.
Endwell New York 13760(US)
Inventor: Shippy, David James
291 Hillside Terrace
Endwell New York 13760(US)

(74) Representative: Jost, Ottokarl, Dipl.-Ing.
IBM Deutschland GmbH Patentwesen und Urheberrecht Schönaicher Strasse 220
D-7030 Böblingen(DE)

(54) **Multiprocessing system with interprocessor communications facility.**

(57) A plurality of processors are connected to the interprocessor communications facility in the multiprocessing system of the invention. The interprocessor communications facility has arbitration circuitry, mailbox circuitry, and processor interrupt circuitry. The interprocessor communications facility of the invention is centralized and does not require the use of main storage. This enables processors to communicate with each other in a fast and efficient manner. The arbitration circuitry prevents simultaneous access of the interprocessor communications facility by more than one processor, and decodes the commands sent from the processors and routes them to the processor interrupt circuitry or to the mailbox circuitry, depending on the command. The mailbox circuitry of the invention receives messages from sending processors and provides them to the intended receiving processors in a safe and secure manner. The processor interrupt circuitry facilitates the interprocessor communications process by handling interprocessor interrupts.

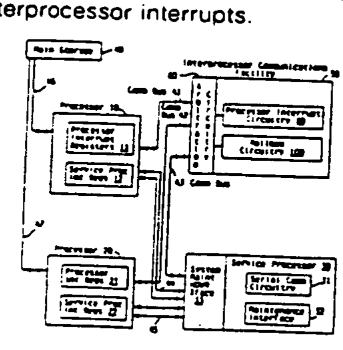


FIG. 1-SYSTEM STRUCTURE

EP 0 376 003 A2

MULTIPROCESSING SYSTEM WITH INTERPROCESSOR COMMUNICATIONS FACILITY

This invention relates to the data processing field. More particularly, this invention is a multiprocessing computer system having an efficient interprocessor communications facility.

In a multiprocessing system, some mechanism usually exists to allow one processor to communicate with another processor. Typically, when one processor wants to send a message to another processor, it places messages and address pointers for interprocessor communications in main storage. Main storage must be accessed by both sending and receiving processors to perform this message transfer. This burdens the main storage and its associated circuitry, delaying other accesses to main storage and reducing overall system performance.

Another problem with the traditional approach of interprocessor communications is the difficulty of communicating with a special type of processor, such as a service processor, when all messages must go through main storage. The service processor performs a variety of diagnostic, maintenance and error recovery operations, and usually cannot write to main storage without quiescing the entire system. Therefore, a service processor could not send a message to another processor without bringing the whole system down, thereby limiting the ability of the service processor to perform error recovery operations.

Prior attempts to improve multiprocessing communications have been complex, inefficient, structured and rigid, and often require all processors to have identical cycle times and operate off of synchronized clock pulses. In addition, these prior attempts have not sufficiently addressed the problem of security and integrity of the interprocessor messages.

It is a principal object of the invention to provide a multiprocessing system with an efficient interprocessor communications facility.

It is another object of the invention to provide a multiprocessing system with an interprocessor communications facility that does not require the use of main storage.

It is another object of the invention to provide a multiprocessing system with an interprocessor communications facility that is flexible enough to be used by a diverse group of processors.

These and other objects are accomplished by the multiprocessing system with an interprocessor communications facility disclosed herein.

A plurality of processors are connected to the interprocessor communications facility in the multiprocessing system of the invention. The interprocessor communications facility has arbitration circuitry, mailbox circuitry, and processor interrupt circuitry. The interprocessor communications facility of the invention is centralized and does not require the use of main storage. This enables processors to communicate with each other in a fast and efficient manner. The arbitration circuitry prevents simultaneous access of the interprocessor communications facility by more than one processor, and decodes the commands sent from the processors and routes them to the processor interrupt circuitry or to the mailbox circuitry, depending on the command. The mailbox circuitry of the invention receives messages from sending processors and provides them to the intended receiving processors in a safe and secure manner. The processor interrupt circuitry facilitates the interprocessor communications process by handling interprocessor interrupts.

The mailbox circuitry contains a mailbox array having a plurality of mailbox entries. Each mailbox entry contains a message field and a lock field. Each lock field contains a lock bit and a lock ID. Each processor has a portion of mailbox entries reserved for its use.

The processors of the multiprocessing system communicate with the interprocessor communications facility of the invention through one or more commands.

The interprocessor communications facility of the invention can be accessed directly by a service processor, so the service processor can communicate with the other processors of the system without quiescing the entire system.

Brief Description of the Drawing

Fig. 1 shows a block diagram of the multiprocessing system of the subject invention.

Fig. 2 shows a block diagram of the interprocessor communications facility of the subject invention.

Fig. 3 shows the arbiter circuitry of the interprocessor communications facility of the subject invention in more detail.

Fig. 4 shows a mailbox entry of the subject invention in more detail.

Fig. 1 shows a block diagram of the multiprocessing system of the subject invention. Processor 10 is connected to interprocessor communications facility 50 via communications bus 41 and to main storage 40

via bus 46. Processor 20 is connected to interprocessor communications facility 50 via communications bus 42 and to main storage 40 via bus 47. Service Processor 30 is connected to interprocessor communications facility 50 via communications bus 43.

In the preferred embodiment, processor 10 is an Input/Output Processing Unit (IOPU), and processor 20 is an Instruction Processing Unit (IPU), although these processors could have other functions and still fall within the scope of the invention. Service Processor 30 performs various diagnostic, maintenance, and error recovery procedures and can initiate Initial Program Load, alter and display General Purpose Registers, alter and display the Program Status Word, alter and display main storage, among other things. Although lines 41-47 are all shown as direct connection busses, any other type of communication path could also be used.

Interrupts between processors 10 and 20 are handled by processor interrupt circuitry 80 of interprocessor communications facility 50 through processor interrupt registers 11 and 21, and will be discussed in more detail later. Due to the special nature of service processor 30, interrupts between service processor 30 and any other processor of the system are typically not handled by processor interrupt circuitry 80 of interprocessor communications facility 50. Instead, interrupts between service processor 30 and any other processor of the system are usually handled by service processor interrupt registers 12 and 22 and maintenance interface 32. In the preferred embodiment service processor interrupt registers 12 and 22 are capable of storing four different interrupt conditions, as shown below in Table 1.

<u>PROCESSOR TO SERVICE PROCESSOR INTERRUPT CONDITIONS</u>

PROCESSOR TO SERVICE PROCESSOR INTERRUPT REQUEST
SERVICE PROCESSOR TO PROCESSOR INTERRUPT RESPONSE
SERVICE PROCESSOR TO PROCESSOR INTERRUPT REQUEST
PROCESSOR TO SERVICE PROCESSOR INTERRUPT RESPONSE

<u>TABLE 1</u>

Service processor 30 interrupts processor 10 via the SERVICE PROCESSOR TO PROCESSOR INTERRUPT REQUEST sent from maintenance interface 32 to service processor interrupt registers 12 via bus 44 and system maintenance hardware interface 33. In the preferred embodiment, system maintenance hardware interface 33 allows service processor 30 to be connected directly to the rest of the multiprocessing system. Service processor 30 then waits for a PROCESSOR TO SERVICE PROCESSOR INTERRUPT RESPONSE from service processor interrupt registers 12 to maintenance interface 32 via bus 44. Service processor 30 interrupts processor 20 in a similar manner, but via bus 45 and using service processor interrupt registers 22.

Processor 10 interrupts service processor 30 via the PROCESSOR TO SERVICE PROCESSOR INTERRUPT REQUEST sent from service processor interrupt registers 12 to maintenance interface 32 via bus 44. Processor 10 then waits for a SERVICE PROCESSOR TO PROCESSOR INTERRUPT RESPONSE from maintenance interface 32 to service processor interrupt registers 12 via bus 44. Processor 20 interrupts service processor 30 in a similar manner, but via bus 45 and using service processor interrupt registers 22.

In the preferred embodiment, interrupts from service processor 30 to any other processor of the system can also be performed by processor interrupt circuitry 80 of interprocessor communications facility 50 in special cases, such as when service processor 30 wants to send an interrupt to all system processors via a single command. The capability to interrupt all processors is a unique capability of processor interrupt circuitry 80 and will be described in more detail later.

Referring again to Fig. 1, interprocessor communications facility 50 contains arbitration circuitry 60, processor interrupt circuitry 80 and mailbox circuitry 100. Communications busses 41-43 provide the inputs to arbitration circuitry 60. Arbitration circuitry 60 is connected to processor interrupt circuitry 80 and mailbox circuitry 100.

Fig. 2 shows a block diagram of interprocessor communications facility 50 of the invention in more detail. Arbitration circuitry 60 prevents simultaneous access of interprocessor communications facility 50 by more than one processor. Arbitration circuitry 60 also decodes the commands sent from the processors and routes them either to processor interrupt circuitry 80 or to mailbox circuitry 100, depending on the command. Arbitration circuitry 60 is made up of holding registers 61-63, arbiter 64, multiplexor 65, and command decoder 66.

When processor 10 wants to communicate with processor 20 in the multiprocessing system, it sends information to holding register 61 via communications bus 41. The information contains a command, an address, and optionally data, such as a message, lock data or interrupt data. When holding register 61 contains information, a request signal is passed to arbiter 64 over request line 71, and the information is presented to the input of multiplexor 65 on data line 74. Arbiter 64 sends a control signal to multiplexor 65 on control line 77, instructing multiplexor 65 to take the information on data line 74 and provide it to command decoder 66. Command decoder 66 decodes the command and sends it either to processor interrupt circuitry 80 or to mailbox circuitry 100.

In the preferred embodiment, each command is composed of 5 bits, thereby permitting 2**5 = 32 commands. The command decoder decodes each command by creating a separate minterm expression for each 5 bit command. A minterm expression is a minimum set of bits that when ANDed together form one of all possibly logically true terms. This decoding operation can be performed by a simple logic circuit arranged using well known digital logic design techniques. The actual design of this circuit is dependent on the coding of the commands chosen by the designer. The logic circuit has one command decode line for each valid command. One command decode line becomes active when the command associated with that line is received by the circuit.

Of the possible 32 commands, 10 are valid or recognized by the decoder, while the remainder are reserved for future use. Any invalid command raises an error signal. The decoder sends out the minterm expressions (signals) to be used by the appropriate facility. A processor interrupt command is sent to processor interrupt circuitry 80. All other commands are sent to mailbox circuitry 100. These commands will be discussed in more detail later.

Fig. 3 shows arbiter 64 of the preferred embodiment in more detail. If there is a signal present on request line 71 from holding register 61, command line 77 is activated, thereby instructing multiplexor 65 to take the information on data line 74 and provide it to command decoder 66. This is done even if there were simultaneously transmitted signals on request lines 72 and/or 73. Therefore, processor 10 is always given priority in arbiter 64 over processors 20 and 30.

If there is a signal present on request line 72 from holding register 62, but no signal on request line 71, command line 78 is activated, thereby instructing multiplexor 65 to take the information on data line 75 and provide it to command decoder 66. This is done even if there was a simultaneously transmitted signal on request line 73. Therefore, processor 20 is always given priority in arbiter 64 over processor 30, but not processor 10.

If there is a signal present on request line 73 from holding register 63, but no signal on request lines 71 or 72, command line 79 is activated, thereby instructing multiplexor 65 to take the information on data line 76 and provide it to command decoder 66. This is done only if there were no simultaneously transmitted signals on request lines 71 or 72.

If it is desired in a particular application to change the priority given to the processors, arbiter 64 can be easily modified by switching lines 71-73 and 77-79 around. Arbiter 64 could also be modified to handle more complex priority schemes.

Referring again to Fig. 2, mailbox circuitry 100 receives messages from sending processors and provides them to the intended receiving processors in a safe and secure manner. Mailbox circuitry 100 is made up of mailbox array 105, message output register 160 and lock output register 161. Mailbox array 105 contains mailbox portions 110, 120 and 130. Mailbox portions 110, 120 and 130 are reserved for the use of processors 10, 20, and 30, respectively, as will be explained in more detail later. Each mailbox portion contains one or more mailbox entries. Each mailbox entry contains message field 140 and lock field 150.

A mailbox entry is shown in more detail in Fig. 4. Message field 140 consists of 8 bytes of data in the preferred embodiment, although this field could be smaller or larger and still fall within the scope of the invention. Usually, these 8 bytes of data would be a message from one processor to another. However, in some instances, such as with asynchronous communications between processors using task control blocks, as will be explained in more detail later, message field 140 is broken up into message field segment 141 and message field segment 142. In this case, segment 141 contains the beginning address of a task control block located in main storage 40. Segment 142 contains the ending address of the task control block.

Lock field 150 contains lock bit 151 and lock ID 152. In the preferred embodiment, lock bit 151 is zero if

the associated mailbox entry is not in use by a processor. If the mailbox entry is in use, lock bit 151 is one, and lock ID field 152 contains identification indicia of the processor using the mailbox entry.

Referring again to Fig. 2, Processor interrupt circuitry 80 facilitates the interprocessor communications process by handling interprocessor interrupts. Processor interrupt circuitry 80 is made up of holding registers 81-83, arbiter 84, multiplexor 85, and output registers 87-88. Arbiter 84 is substantially the same as arbiter 64, shown in more detail in Fig. 3, as has been discussed.

When command decoder 66 decodes a processor interrupt command, it sends the command on to processor interrupt circuitry 80. Note that processor interrupt circuitry 80 is substantially similar to arbitration circuitry 60. This is done to provide an additional holding area for processor interrupt commands, since processor interrupt commands can take longer to execute than the other commands which are sent to mailbox circuitry 100.

In the preferred embodiment, command decoder 66 uses four clock cycles, T0, T1, T2, and T3, to decode and execute a command. During T0, the command is decoded by command decoder 66 and one of the command decode lines becomes active. From here on, the exact sequence of control signals depends on the command being executed. For example, when a READ MESSAGE command is decoded, the address of the mailbox entry is gated to mailbox array 105. The array will access the addressed location and the message will be sent to output register 160, from which it is returned to the requesting processor. These steps are performed during clock cycles T1-T3.

The command decode lines are used to gate the T-clocks through the control logic to determine which operations are performed and when. For example, when the PROCESSOR INTERRUPT command is decoded, it will gate the T2 signal to the proper holding register (81, 82, or 83) to latch the address and data. Some operations are performed by more than one command, so several command decode lines may be ORed together to gate a T-clock. For example, several commands cause the lock field of a mailbox entry to be read. Those commands are joined together to read the lock field and latch the output data into output register (161).

The commands available for use by the processors of the multiprocessing system to facilitate interprocessor communications are shown below in Table 2.

MAILBOX CIRCUITRY COMMANDS

WRITE MESSAGE
READ MESSAGE
READ LOCK
WRITE LOCK
TEST AND SET LOCK
RESET LOCK
TEST AND SET LOCK AND READ MESSAGE
RESET LOCK AND WRITE MESSAGE
TEST AND SET LOCK AND WRITE MESSAGE

PROCESSOR INTERRUPT CIRCUITRY COMMAND

PROCESSOR INTERRUPT

TABLE 2

MAILBOX CIRCUITRY COMMANDS

Message Commands

The WRITE MESSAGE command is used by a processor who wants to send a message to another processor in the system. This command contains the message intended for another processor. In the preferred embodiment, this message can be up to eight bytes long, but this length could be different and still fall within the scope of the invention. The WRITE MESSAGE command also contains an address of a mailbox entry in mailbox array 105. This mailbox entry is within the message array portion reserved for the processor for which the message is intended. When this command is received by command decoder 66, the message is placed in message field 140 of the addressed mailbox entry.

The READ MESSAGE command is used by a processor who wants to read a message placed in one of its mailbox entries by another processor. This command contains the address of the mailbox entry that has the message the processor wants to read. When this command is received by command decoder 66, the message is retrieved from the addressed mailbox entry and placed in message output register 160, where it is sent on return line 49 to the processor who issued the command.

Lock Commands

The READ LOCK command is used by a processor who wants to read the data contained in the lock field of a mailbox entry. This command contains the address of the mailbox entry corresponding to the lock field the processor wants to read. When this command is received by command decoder 66, the data contained in lock field 150 of the addressed mailbox entry is retrieved from the addressed mailbox entry and placed in lock output register 161, where it is sent on return line 49 to the processor who issued the command.

The WRITE LOCK command is used by a processor who wants to write data to the lock field of a mailbox entry. This command contains the data the processor wants to write, along with the address of the mailbox entry corresponding to the lock field the processor wants to write. When this command is received by command decoder 66, the data is placed in the addressed mailbox entry in lock field 150.

If there is data in the array with bad parity, such as during power up, this command can be used to initialize the array.

The TEST AND SET LOCK command is used by a processor who wants to check a lock field of a mailbox entry to see if the mailbox entry is in use, and, if not, write data to the lock field. This command contains the data the processor wants to write, along with the address of the mailbox entry corresponding to the lock field the processor wants to write. When this command is received by command decoder 66, the most significant bit of the addressed lock field is checked. If the most significant bit of the lock field is zero, indicating that the mailbox entry is not in use, then the data from the processor is written into the lock field of the specified mailbox entry. This data in the preferred embodiment would be identification indicia of the processor making the request. By inserting this data in the lock field (and setting the most significant bit to one), the processor indicates that it is using this mailbox entry. If the most significant bit of the lock field is one, the write does not take place. In both cases, the data contained in the lock field before the command was received is placed in lock output register 161, where it is sent on return line 49 to the processor who issued the command.

The RESET LOCK command is used by a processor who wants to indicate that it no longer is using a mailbox entry. This command contains the address of the mailbox entry no longer needed. When this command is received by command decoder 66, the most significant bit of the lock field corresponding to the addressed mailbox entry is reset to zero, indicating that the mailbox entry is not in use. If that bit were already a zero, no error is flagged. By convention, a processor would only use this command for mailbox entries it was using.

Combination Commands

In order to increase the efficiency of using these commands, combination commands made up of the commands shown above can be used by the processors of the multiprocessing system to facilitate interprocessor communications.

The TEST AND SET LOCK AND READ MESSAGE is used by a processor who wants to use both the TEST AND SET LOCK command and the READ MESSAGE command, as described above.

The RESET LOCK AND WRITE MESSAGE is used by a processor who wants to use both the RESET LOCK command and the WRITE MESSAGE command, as described above.

The TEST AND SET LOCK AND WRITE MESSAGE is used by a processor who wants to use both the TEST AND SET LOCK command and the WRITE MESSAGE command, as described above. Neither the lock write nor the message write takes place if the lock field indicates that the mailbox entry is in use.

PROCESSOR INTERRUPT CIRCUITRY COMMAND

The PROCESSOR INTERRUPT command is used by a processor that wants to set or reset a processor interrupt, including its own interrupt. This command contains the address of the targeted processor or processors and one byte of interrupt data that sets or resets a processor interrupt. In the preferred embodiment, this address is encoded to provide more function and flexibility. Encoding is used to compress the address field so that enough addresses are provided to address processors individually or as groups of processors. For example, address 1000 is processor 10 and address 1001 is processor 20. Address 1111 is the address for "All Processors" which includes both processor 10 and processor 20. When such a 'broadcast' address is used, processor interrupt circuitry 80 must send the data byte to all processors in that group and will wait until all have responded.

If, for example, processor 10 wants to send a byte of interrupt data to processor 20, it sends a processor interrupt command along with the address "1001" to holding register 61. Arbitration circuitry 60 presents the command to command decoder 66, who decodes the command and sends it and the interrupt data byte on to holding register 81 in processor interrupt circuitry 80. Processor interrupt circuitry 80 operates in a manner similar to arbitration circuitry 60, as has been discussed, and eventually places the interrupt byte in output register 88, the output register dedicated to processor 20. The interrupt byte is then sent on return line 49 back to processor 20.

Note that service processor 30 has a separate interrupt facility with processors 10 and 20 via service processor interfaces 12 and 22, as has been discussed, and is not addressable by processor interrupt circuitry 80. However, service processor 30 is able to originate a processor interrupt command and would do so in special cases, such as when it wants to send an interrupt to all system processors via a single command.

Although only processor 10 and processor 20 are shown in Fig. 1 as being addressable by processor interrupt circuitry 80, several additional processors, such as I/O processors, Instruction processors, or other type of processors, could be part of the multiprocessing system and could be addressable by processor interrupt circuitry 80. Any particular processor could be assigned to more than one group if desired. For example, one group could be implemented which includes "all processors" while another could be "all I/O processors". In this example, each I/O processor would be in both groups. If additional processors are added, corresponding additional holding registers and output registers would be added, and arbiters 64 and 84 would be modified slightly in a manner known to those skilled in the art.

Several examples of how the above commands can be used in the multiprocessing system of the invention will now be discussed.

Processor 10 to Processor 20 Synchronous

This example shows how the invention can be used to allow processor 10 to synchronously send a message to processor 20, and have processor 20 respond to the synchronous message, while processor 10 waits for the response from processor 20. Processor 10 issues a TEST AND SET LOCK AND WRITE MESSAGE command to the address of a mailbox entry in mailbox portion 120 reserved for synchronous communication with processor 20, and checks that the lock was obtained. If it was obtained, the data that was written to the mailbox entry contains the message information and the lock field contains the lock ID of processor 10. If the lock was not obtained, then processor 10 must wait for the mailbox entry to be unlocked. In addition, the lock bit is set to one, indicating that the mailbox entry is in use, thereby preventing all other processors from using synchronous communication with processor 20 until processor 10 has released the lock.

Next, processor 10 sends an interrupt to processor 20 indicating a synchronous message is in the mailbox waiting for processing. Processor 10 uses the PROCESSOR INTERRUPT command to set the

synchronous interrupt bit in processor 20. Processor 20 will soon detect the interrupt and will realize that a message is waiting in the mailbox. Processor 20 then will use a READ MESSAGE command to get the message. Processor 20 then uses the PROCESSOR INTERRUPT command to reset the interrupt that is being processed. Processor 20 dedicates its resources to process this message, and places the message response back in the mailbox array at the address of a mailbox entry located in mailbox array portion 110 reserved for synchronous responses to processor 10 with a WRITE MESSAGE command. Since the message handling is now complete, processor 20 indicates this with a PROCESSOR INTERRUPT command to set the response interrupt bit in processor 10. Since processor 20 has completed the message handling, it can now proceed with other tasks.

Meanwhile, processor 10 detects the response interrupt bit, signalling that processor 20 has completed the message handling and the response is in the mailbox array. Processor 10 obtains the response by executing a READ MESSAGE command followed by a RESET LOCK command. The message response is analyzed as desired, and the process is complete. The synchronous communications resources for processor 20 are now available once again to any processor.

This example uses a number of mailbox commands and processor interrupt commands to handle a message and message response in multiple processors. Main storage does not need to be accessed to accomplish the communications in both directions, thereby resulting in a significant performance advantage.

Processor 10 to Processor 20 Asynchronous

Asynchronous communications is accomplished by defining a list of task control blocks stored in main storage 40 for each processor receiving asynchronous communication. Instead of an eight byte message being placed in message field 140 of a mailbox entry (Fig. 4), task control block header information is used. Specifically, message field segment 141 contains the beginning address of the task control block in main storage 40, and message field 142 contains the ending address of the task control block.

When processor 10 finds a task that needs to be executed asynchronously by processor 20, the task is put into a task control block in main memory 40. Processor 10 then issues a TEST AND SET LOCK AND READ MESSAGE command to the address of the mailbox entry reserved for asynchronous communications with processor 20. Processor 10 then checks if the lock has been obtained, and if so, uses the header data. Processor 10 then enqueues the task control block on processor 20's list, and uses the PROCESSOR INTERRUPT command to notify processor 20 if this enqueue is the first entry on the list. If this is not the first entry, the interrupt has already been set by convention. The list header is then updated in the mailbox entry and the lock reset with a RESET LOCK AND WRITE MESSAGE command. This completes processor 10's participation in asynchronous communication and frees up resources so that other processors may enqueue or dequeue tasks on processor 20's list.

When processor 20 detects the interrupt for asynchronous communication, the task control block header contained in the mailbox entry is again accessed by a TEST AND SET AND READ MESSAGE command. If the lock is obtained, processor 20 is allowed to dequeue list entries from the linked list. If it is not obtained, it can not dequeue asynchronous tasks.

When the lock is obtained, processor 20 dequeues the top entry from the list. If it is removing the last entry in its own list, the asynchronous communications interrupt is reset by using the PROCESSOR INTERRUPT command and addressing it's own interrupt bit. This must be done while the mailbox lock is held by processor 20 to maintain integrity of the list.

The task control block header is updated and the lock reset by using the RESET LOCK AND WRITE MESSAGE command. This completes processor 20's participation in the asynchronous communication and frees up the resources for other processors to enqueue more tasks on processor 20's list while it executes the task it just removed from the list.

This example uses a few mailbox commands in conjunction with processor interrupt commands to allow many processors to access individual work lists and pass work from one to another quickly and efficiently.

Service Processor 30 to Processor 20

The following shows how the invention could be used to satisfy an operator request to display main storage 40 starting at a specified address. The operator sends a request to the service processor 30, via keyboard input, to display sixty four bytes of main storage starting from a specified address. Service processor 30 first writes eight bytes of data to the message field of a mailbox entry in mailbox array portion

120 reserved for service processor 30 to processor 20 communications using the WRITE MESSAGE command. This data will include a predefined command instructing processor 20 to read from main storage 40, along with the main storage address.

Service processor 30 then sends a SERVICE PROCESSOR TO PROCESSOR INTERRUPT REQUEST from maintenance interface 32 to service processor interrupt registers 22 over bus 45 to processor 20. Service processor 30 then waits for a PROCESSOR TO SERVICE PROCESSOR INTERRUPT RESPONSE from processor 20 over bus 45.

When processor 20 detects the SERVICE PROCESSOR TO PROCESSOR INTERRUPT REQUEST, processor 20 issues a READ MESSAGE command to read the message contained in the mailbox entry. Processor 20 then inspects the message and deciphers that service processor 30 is requesting sixty four bytes of data starting from a specified main storage address.

Processor 20 reads the first eight bytes of data starting at the main storage address. It then transfers this data to the message field of the first mailbox entry in mailbox array portion 130 reserved for the use of service processor 30 via a WRITE MESSAGE command. In the preferred embodiment, service processor 30 has several mailbox entries reserved for its use in mailbox array portion 130 to efficiently handle data transfers such as that described in this example.

Processor 20 then reads the next eight bytes from main storage 40 and writes this data to the message field of the second mailbox entry reserved for the use of service processor 30 in mailbox array portion 130 via another WRITE MESSAGE command. This process continues until all sixty four bytes have been read from main storage and transferred to subsequent mailbox entries in mailbox array portion 130.

Processor 20 then responds to service processor 30 by sending a PROCESSOR TO SERVICE PROCESSOR INTERRUPT RESPONSE via bus 45. Service processor 30 then reads the data contained in the message fields of its mailbox entries using several READ MESSAGE commands. Finally, service processor 30 outputs the requested sixty four bytes of data on the operator console.

Claims

1. A multiprocessing computer system having efficient interprocessor communications, comprising:
a first processor;
a second processor;
interprocessor communications means, connected to said first processor and said second processor, for controlling communications between said first processor and said second processor directly without the use of main storage, said means further comprising:
arbitration means, connected to said first processor and said second processor, for permitting only one processor to access said interprocessor communications means at a time; and
mailbox means, connected to said arbitration means, for delivering interprocessor messages between said first processor and said second processor.

2. The multiprocessor computer system of claim 1, wherein said arbitration means further comprises:
a first holding register, connected to said first processor, for holding first information from said first processor;
a second holding register, connected to said second processor, for holding second information from said second processor;
a first multiplexor, connected to said first holding register by a first data line, and connected to said second holding register by a second data line;
a first arbiter, connected to said first holding register by a first request line, connected to said second holding register by a second request line and connected to said first multiplexor by a control line;
said first holding register sending a first signal to said first arbiter over said first request line when it is holding said first information; and
said second holding register sending a second signal to said first arbiter over said second request line when it is holding said second information.

3. The multiprocessor computer system of claim 2, wherein said arbitration means further comprises:
a command decoder, connected between said first multiplexor and said mailbox means, for decoding a first command from said first information and a second command from said second information.

4. The multiprocessing system of claim 3, wherein said first arbiter further comprises:
means for instructing said first multiplexor to send said first information contained on said first data line to said command decoder responsive to said first signal.

5. The multiprocessing system of claim 4, wherein said first arbiter further comprises:

means for instructing said first multiplexor to send said second information contained on said second data line to said command decoder responsive to said second signal if said second signal is received before said first signal.

6. The multiprocessor computer system of claim 1, wherein said mailbox means further comprises: a mailbox array having a plurality of mailbox entries, wherein a first portion of said mailbox entries are reserved for said first processor and a second portion of said mailbox entries are reserved for said second processor.

7. The multiprocessor computer system of claim 6, wherein each of said mailbox entries comprises: a message field for storing interprocessor messages; and a lock field for storing lock information about said mailbox entry.

8. The multiprocessor computer system of claim 7, wherein said lock field comprises: a lock bit for indicating if said mailbox entry is in use; and a lock ID for indicating which processor is using said mailbox entry.

9. The multiprocessor computer system of claim 5, wherein said mailbox means further comprises: a mailbox array having a plurality of mailbox entries, wherein a first portion of said mailbox entries are reserved for said first processor and a second portion of said mailbox entries are reserved for said second processor.

10. The multiprocessor computer system of claim 9, wherein each of said mailbox entries comprises: a message field for storing interprocessor messages; and a lock field for storing lock information about said mailbox entry.

11. The multiprocessor computer system of claim 10, wherein said lock field comprises: a lock bit for indicating if said mailbox entry is in use; and a lock ID for indicating which processor is using said mailbox entry.

12. The multiprocessor computer system of claim 3, wherein said interprocessor communications means further comprises: processor interrupt means, connected to said arbitration means and said mailbox means, for delivering interprocessor interrupts between said first processor and said second processor.

13. The multiprocessor computer system of claim 11, wherein said interprocessor communications means further comprises: processor interrupt means, connected to said arbitration means and said mailbox means, for delivering interprocessor interrupts between said first processor and said second processor.

14. The multiprocessor computer system of claim 13, wherein said first information comprises: a first command; a first address; and a first data.

15. The multiprocessor computer system of claim 14, wherein said second information comprises: a second command; a second address; and a second data.

16. The multiprocessing computer system of claim 15, wherein said command decoder further comprises: means for routing said first data to the mailbox entry specified by said first address if said first command is a mailbox command.

17. The multiprocessing computer system of claim 16, wherein said command decoder further comprises: means for routing said first data to said processor interrupt means if said first command is a processor interrupt command.

18. The multiprocessor computer system of claim 17, wherein said processor interrupt means further comprises: a fourth holding register, connected to said arbitration means, for holding said first data from said first processor; a fifth holding register, connected to said arbitration means, for holding said second data from said second processor; a second multiplexor, connected to said fourth holding register by a fourth data line, and connected to said fifth holding register by a fifth data line; and a second arbiter, connected to said fourth holding register by a fourth request line, connected to said fifth holding register by a fifth request line, and connected to said second multiplexor by a second control line: said fourth holding register sending a fourth signal to said second arbiter over said fourth request line when

it is holding said first data,
said fifth holding register sending a fifth signal to said second arbiter over said fifth request line when it is holding said second data,
said second arbiter comprising means for instructing said second multiplexor to send said first data contained on said fourth data line to the processor specified in said first address responsive to said fourth signal.

19. The multiprocessor computer system of claim 18, wherein said processor interrupt means further comprises:
said second arbiter comprising means for instructing said second multiplexor to send said second data contained on said fifth data line to the processor specified in said second address responsive to said fifth signal if said fifth signal is received before said fourth signal.

20. A multiprocessing computer system of claim 1 having efficient interprocessor communications, a first processor and a second processor comprising:
a service processor capable of performing diagnostic and error recovery procedures for said system:
interprocessor communications means, connected to said first processor, said second processor, and said service processor for controlling communications between said first processor, said second processor, and said service processor directly without the use of main storage, said means further comprising:
arbitration means, connected to said first processor, said second processor, and said service processor for permitting only one processor to access said interprocessor communications means at a time; and
mailbox means, connected to said arbitration means, for delivering interprocessor messages between said first processor, said second processor, and said service processor.

21. The multiprocessor computer system of claim 20, wherein said arbitration means further comprises:
a first holding register, connected to said first processor, for holding first information from said first processor;
a second holding register, connected to said second processor, for holding second information from said second processor;
a third holding register, connected to said service processor, for holding third information from said service processor;
a first multiplexor, connected to said first holding register by a first data line, connected to said second holding register by a second data line, and connected to said third holding register by a third data line:
a first arbiter, connected to said first holding register by a first request line, connected to said second holding register by a second request line, connected to said third holding register by a third request line, and connected to said first multiplexor by a first control line;
said first holding register sending a first signal to said first arbiter over said first request line when it is holding said first information;
said second holding register sending a second signal to said first arbiter over said second request line when it is holding said second information; and
said third holding register sending a third signal to said first arbiter over said third request line when it is holding said third information.

22. The multiprocessor computer system of claim 21, wherein said arbitration means further comprises:
a command decoder, connected between said first multiplexor and said mailbox means, for decoding a first command from said first information, a second command from said second information, and a third command from said third information.

23. The multiprocessing system of claim 22, wherein said first arbiter further comprises:
means for instructing said first multiplexor to send said first information contained on said first data line to said command decoder responsive to said first signal.

24. The multiprocessing system of claim 23, wherein said first arbiter further comprises:
means for instructing said first multiplexor to send said second information contained on said second data line to said command decoder responsive to said second signal if said second signal is received before said first signal.

25. The multiprocessing system of claim 24, wherein said first arbiter further comprises:
means for instructing said first multiplexor to send said third information contained on said third data line to said command decoder responsive to said third signal if said third signal is received before said first signal and said second signal.

26. The multiprocessor computer system of claim 20, wherein said mailbox means further comprises:
a mailbox array having a plurality of mailbox entries, wherein a first portion of said mailbox entries are reserved for said first processor, a second portion of said mailbox entries are reserved for said second processor, and a third portion of said mailbox entries are reserved for said service processor.

27. The multiprocessor computer system of claim 26, wherein each of said mailbox entries comprises:
a message field for storing interprocessor messages; and
a lock field for storing lock information about said mailbox entry.

28. The multiprocessor computer system of claim 27, wherein said lock field comprises:
a lock bit for indicating if said mailbox entry is in use; and
a lock ID for indicating which processor is using said mailbox entry.

29. The multiprocessor computer system of claim 25, wherein said mailbox means further comprises:
a mailbox array having a plurality of mailbox entries, wherein a first portion of said mailbox entries are reserved for said first processor, a second portion of said mailbox entries are reserved for said second processor, and a third portion of said mailbox entries are reserved for said service processor.

30. The multiprocessor computer system of claim 29, wherein each of said mailbox entries comprises:
a message field for storing interprocessor messages; and
a lock field for storing lock information about said mailbox entry.

31. The multiprocessor computer system of claim 30, wherein said lock field comprises:
a lock bit for indicating if said mailbox entry is in use; and
a lock ID for indicating which processor is using said mailbox entry.

32. The multiprocessor computer system of claim 23, wherein said interprocessor communications means further comprises:
processor interrupt means, connected to said arbitration means and said mailbox means, for delivering interprocessor interrupts between said first processor and said second processor.

33. The multiprocessor computer system of claim 31, wherein said interprocessor communications means further comprises:
processor interrupt means, connected to said arbitration means and said mailbox means, for delivering interprocessor interrupts between said first processor and said second processor.

34. The multiprocessor computer system of claim 33, wherein said first information comprises:
a first command;
a first address; and
a first data.

35. The multiprocessor computer system of claim 34, wherein said second information comprises:
a second command;
a second address; and
a second data.

36. The multiprocessor computer system of claim 35, wherein said third information comprises:
a third command;
a third address; and
a third data.

37. The multiprocessing computer system of claim 36, wherein said command decoder further comprises:
means for routing said first data to the mailbox entry specified by said first address if said first command is a mailbox command.

38. The multiprocessing computer system of claim 37, wherein said command decoder further comprises:
means for routing said first data to said processor interrupt means if said first command is a processor interrupt command.

39. The multiprocessor computer system of claim 38, wherein said processor interrupt means further comprises:
a fourth holding register, connected to said arbitration means, for holding said first data from said first processor;
a fifth holding register, connected to said arbitration means, for holding said second data from said second processor;
a sixth holding register, connected to said arbitration means, for holding said third data from said service processor;
a second multiplexor, connected to said fourth holding register by a fourth data line, connected to said fifth holding register by a fifth data line, and connected to a sixth holding register by a sixth data line;
a second arbiter, connected to said fourth holding register by a fourth request line, connected to said fifth holding register by a fifth request line, connected to said sixth holding register by a sixth request line, and connected to said second multiplexor by a second control line; and
said fourth holding register sending a fourth signal to said second arbiter over said fourth request line when

it is holding said first data.
said fifth holding register sending a fifth signal to said second arbiter over said fifth request line when it is holding said second data,
said sixth holding register sending a sixth signal to said second arbiter over said sixth request line when it is holding said third data,
said second arbiter comprising means for instructing said second multiplexor to send said first data contained on said fourth data line to the processor specified in said first address responsive to said fourth signal.

40. The multiprocessor computer system of claim 39, wherein said processor interrupt means further comprises:
said second arbiter comprising means for instructing said second multiplexor to send said second data contained on said fifth data line to the processor specified in said second address responsive to said fifth signal if said fifth signal is received before said fourth signal.

41. The multiprocessor computer system of claim 40, wherein said processor interrupt means further comprises:
said second arbiter comprising means for instructing said second multiplexor to send said third data contained on said sixth data line to the processor specified in said third address responsive to said sixth signal if said sixth signal is received before said fourth signal and said fifth signal.

42. The multiprocessor computer system of claim 41, wherein said first command is a write message command.

43. The multiprocessor computer system of claim 41, wherein said first command is a read message command.

44. The multiprocessor computer system of claim 41, wherein said first command is a write lock command.

45. The multiprocessor computer system of claim 41, wherein said first command is a read lock command.

46. The multiprocessor computer system of claim 41, wherein said first command is a test and set lock command.

47. The multiprocessor computer system of claim 41, wherein said first command is a reset lock command.

48. The multiprocessor computer system of claim 41, wherein said first command is a test and set lock and read message command.

49. The multiprocessor computer system of claim 41, wherein said first command is a reset lock and write message command.

50. The multiprocessor computer system of claim 41, wherein said first command is a test and set lock and write message command.

51. A method of communication between a first processor and a second processor in a multiprocessing system having interprocessor communications means connected between said first processor and said second processor, said interprocessor communications means having a first mailbox entry for holding messages for said first processor and having a second mailbox entry for holding messages for said second processor, said method comprising the steps of:
said first processor sending a write message command to said interprocessor communications means, said command containing a message for said second processor and specifying the address of said second mailbox entry;
said interprocessor communications means placing said message for said second processor in said second mailbox entry;
said second processor transmitting a read message command to said interprocessor communications means, said command specifying the address of said second mailbox entry; and
said interprocessor communications means sending said message for said second processor to said second processor responsive to said transmitting step.

52. A method of claim 51 with each of said mailbox entries having a message field and a lock field, said method comprising the steps of:
said first processor sending a test and set lock and write message command to said interprocessor communications means, said command containing a message for said second processor and specifying the address of said second mailbox entry;
said interprocessor communications means testing the lock field of said second mailbox entry to determine if said second mailbox entry is available for use;
said interprocessor communications means placing identification indicia of said first processor into said lock

field of said second mailbox entry responsive to said testing step indicating said second mailbox entry was available for use;
said interprocessor communications means placing said message for said second processor in said second mailbox entry responsive to said testing step indicating said second mailbox entry was available for use;
said second processor transmitting a read message command to said interprocessor communications means, said command specifying the address of said second mailbox entry; and
said interprocessor communications means sending said message for said second processor to said second processor responsive to said transmitting step.

53. A method of claim 52 said interprocessor communications means also having processor interrupt means, said method comrpising the steps of:
said first processor sending a test and set lock and write message command to said interprocessor communications means, said command containing a message for said second processor and specifying the address of said second mailbox entry;
said interprocessor communications means testing the lock field of said second mailbox entry to determine if said second mailbox entry is available for use;
said interprocessor communications means placing identification indicia of said first processor into said lock field of said second mailbox entry responsive to said testing step indicating said second mailbox entry was available for use;
said interprocessor communications means placing said message for said second processor in said second mailbox entry responsive to said testing step indicating said second mailbox entry was available for use;
said first processor sending a processor interrupt command to said interprocessor communications means, said processor interrupt command containing an interrupt for said second processor;
said interprocessor communications means sending said interrupt for said second processor to said second processor::
said second processor transmitting a read message command to said interprocessor communications means, said command specifying the address of said second mailbox entry, said transmitting step responsive to said interrupt for said second processor; and
said interprocessor communications means sending said message for said second processor to said second processor responsive to said transmitting step.

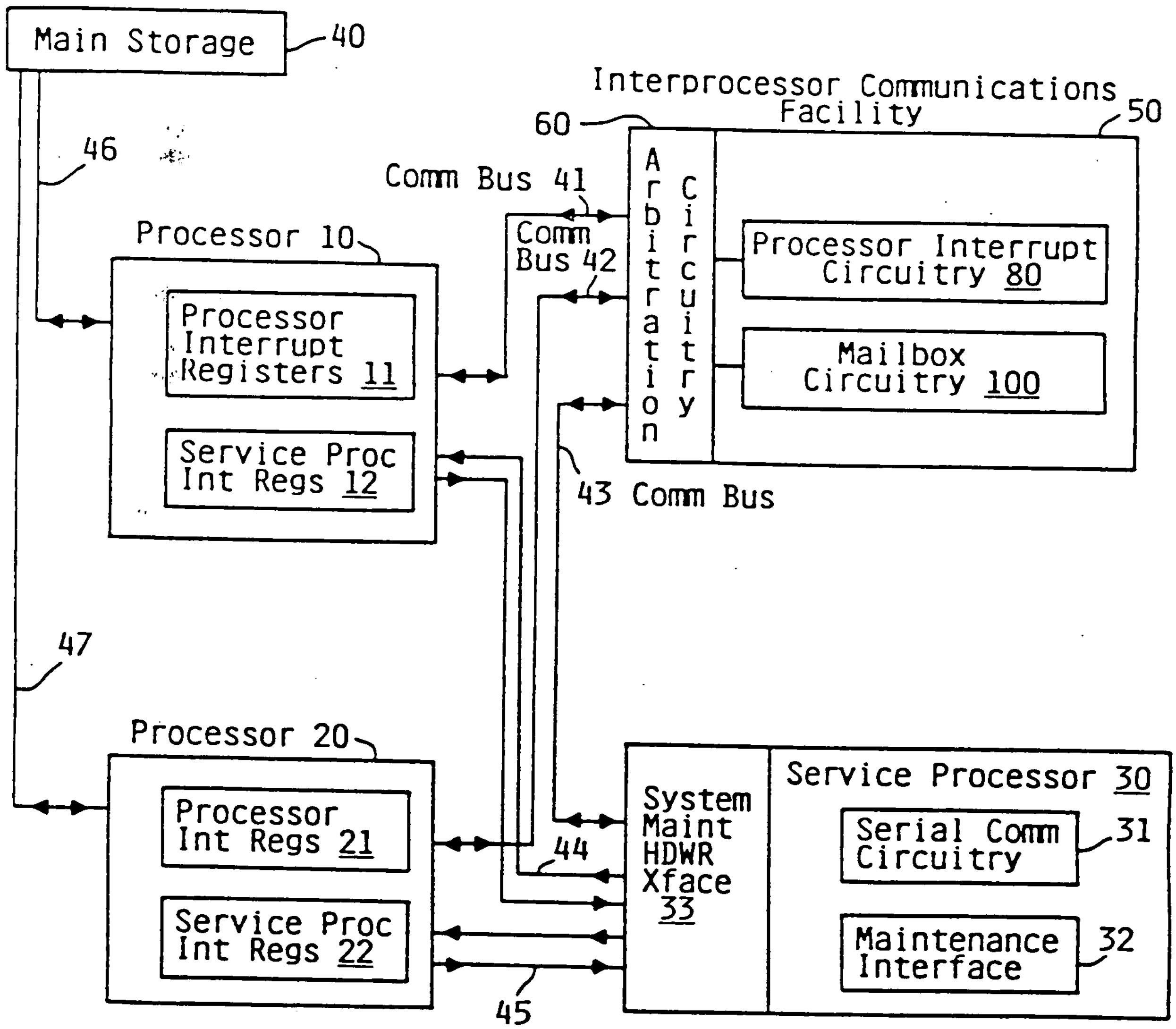
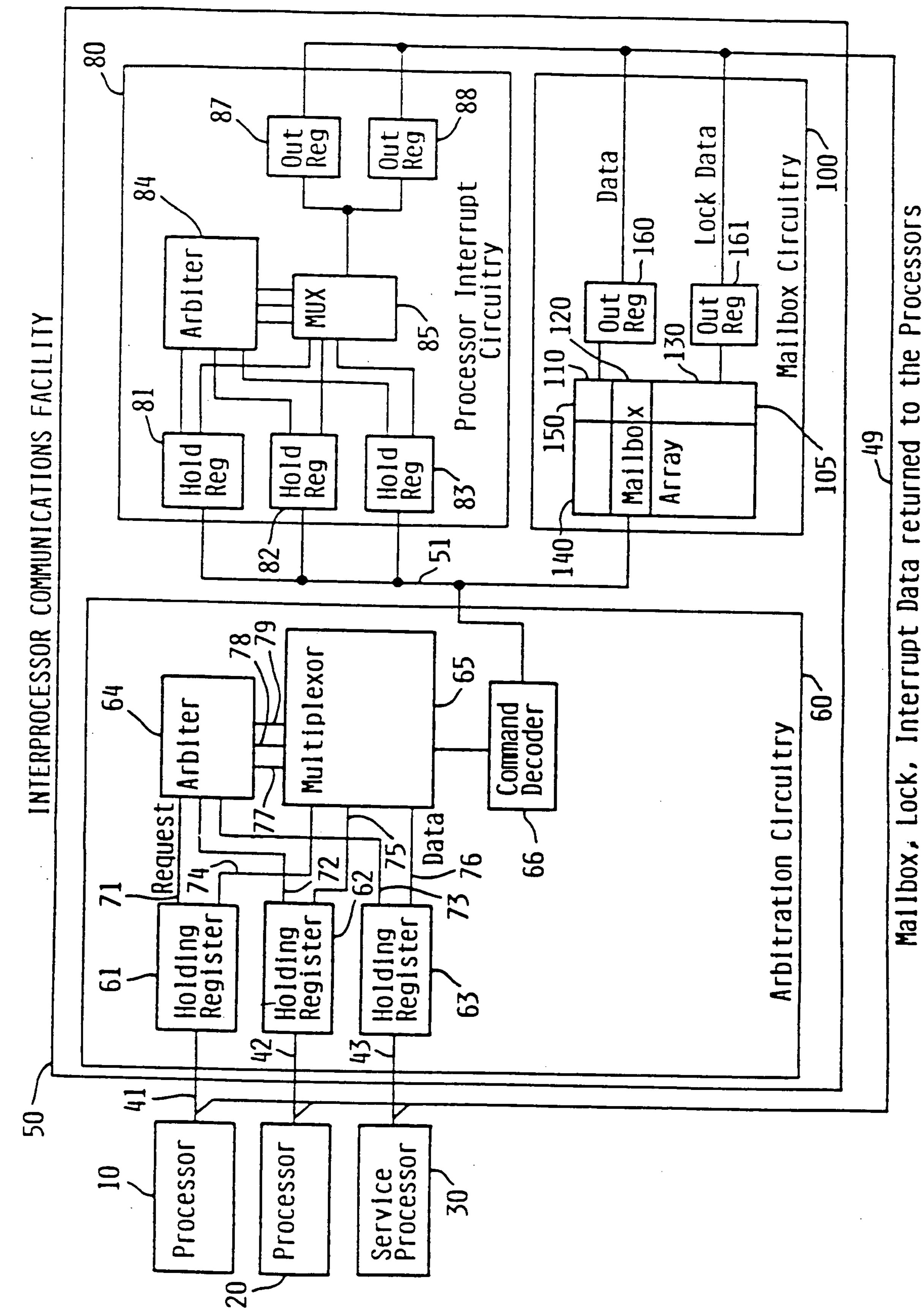


FIG. 1 - SYSTEM STRUCTURE



INTERPROCESSOR COMMUNICATIONS FACILITY
Processor
Processor
Service Processor
Holding Register
Holding Register
Holding Register
Arbiter
Multiplexor
Command Decoder
Request
Data
Arbitration Circuitry
Hold Reg
Hold Reg
Hold Reg
Arbiter
MUX
Out Reg
Out Reg
Processor Interrupt Circuitry
Mailbox
Array
Out Reg
Out Reg
Data
Lock Data
Mailbox Circuitry
Mailbox, Lock, Interrupt Data returned to the Processors
10
20
30
41
42
43
49
50
51
60
61
62
63
64
65
66
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
87
88
100
105
110
120
130
140
150
160
161

FIG. 2

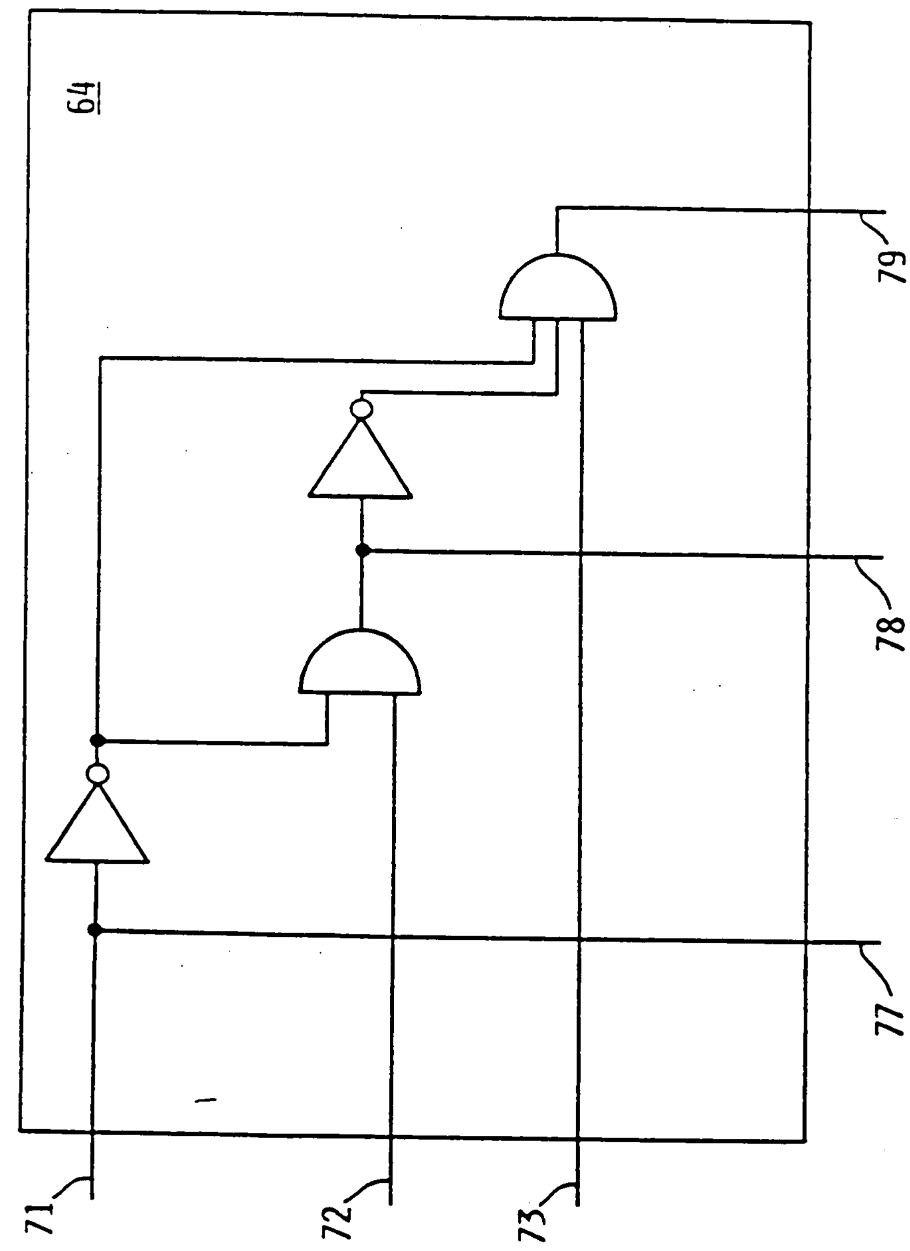


FIG. 3

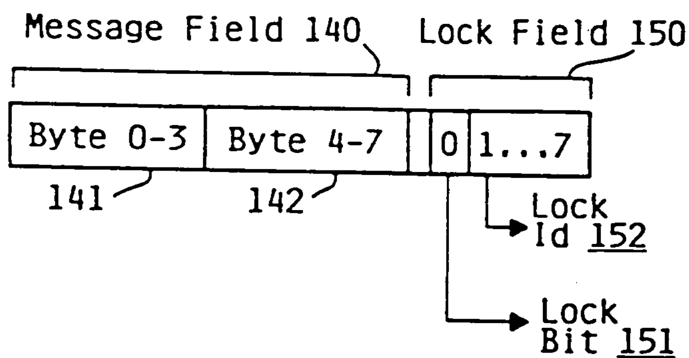


FIG. 4 - MAILBOX ENTRY

Best Available Copy

#7

European
Patent Office

DOCUMENTS CONSII

Category	Citation of document with ... of relev...
Y	US-A-4 214 305 (TOKITA E * the whole document *
Y A	EP-A-0 201 020 (BULL HN I ITALIA S.P.A.) * the whole document *
A	ELECTRONICS & WIRELESS WORLD. vol. 9 SUTTON GB pages 875 - 881 "Multiprocessor systems" * the whole document *
A	US-A-4 698 753 (HUBBINS I * the whole document *
A	IBM TECHNICAL DISCLOSUF May 1986, NEW YORK US pa "Communication mechanism t and a processor" * the whole document *
A	EP-A-0 197 499 (HONEYWEI * the whole document *
A	EP-A-0 029 975 (HONEYWEI ITALIA S.P.A) * the whole document *
A	GB-A-2 091 917 (ATEX INC.) * the whole document *

The present search report has been

Place of search	
Berlin	

CATEGORY OF CITED DOCUMENT
X : particularly relevant if taken alone
Y : particularly relevant if combined with ano document of the same category
A : technological background
O : non-written disclosure
P : intermediate document
T : theory or principle underlying the inventic

Office européen des brevets

Publi

(12) EUROPEAN PATENT AP

(21) Application number: 89122414.9

(51) Int. C

(22) Date of filing: 05.12.89

(30) Priority: 29.12.88 US 291890

(43) Date of publication of application:
04.07.90 Bulletin 90/27

(84) Designated Contracting States:
DE FR GB

(88) Date of deferred publication of the search report:
18.12.91 Bulletin 91/51

(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: **Bolan, Timothy Vincent**
2922 Ricky Dr.
Endwell, NY 13760(US)
Inventor: **Boston, Josephine Ann**
206 Prospect Street

End
Inve
931
End
Inve
300
Enc
Inve
103
Enc
Inve
292
En
Inv
29
En

(74) Re
IBI
Ur
W

(54) Multiprocessing system with interprocessor communic

(57) A plurality of processors are connected to the interprocessor communications facility in the multiprocessing system of the invention. The interprocessor communications facility has arbitration circuitry, mailbox circuitry, and processor interrupt circuitry. The interprocessor communications facility of the invention is centralized and does not require the use of main storage. This enables processors to communicate with each other in a fast and efficient manner. The arbitration circuitry prevents simultaneous access of the interprocessor communications facility by more than one processor, and decodes the commands sent from the processors and routes them to the processor interrupt circuitry or to the mailbox circuitry, depending on the command. The mailbox circuitry of the invention receives messages from sending processors and provides them to the intended receiving processors in a safe and secure manner. The processor interrupt circuitry facilitates the interprocessor communications process by han-

dlin

EP 0 376 003 A3

G06F15/1632

Office européen des brevets

Publication nu

EUROPEAN PATENT APPLICA

Application number: 89122414.9

Int. Cl.5: G06

Date of filing: 05.12.89

Priority: 29.12.88 US 291890

Date of publication of application:
04.07.90 Bulletin 90/27

Designated Contracting States:
DE FR GB

Date of deferred publication of the search report:
18.12.91 Bulletin 91/51

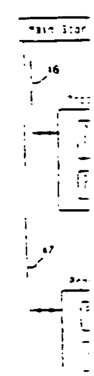
Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)

Inventor: **Bolan, Timothy Vincent**
2922 Ricky Dr.
Endwell, NY 13760(US)
Inventor: **Boston, Josephine Ann**
206 Prospect Street
Endicott, NY
Inventor: **Han**
931 Byford E
Endwell, NY
Inventor: **Lau**
300 Princetc
Endicott, NY
Inventor: **Rin**
1036 Schuyl
Endicott, NY
Inventor: **Rur**
2926 Twiligh
Endwell Nev
Inventor: **Shi**
291 Hillside
Endwell Ne

Representati
IBM Deutsc
Urheberrec
W-7030 Böb

Multiprocessing system with interprocessor communications fa

A plurality of processors are connected to the interprocessor communications facility in the multiprocessing system of the invention. The interprocessor communications facility has arbitration circuitry, mailbox circuitry, and processor interrupt circuitry. The interprocessor communications facility of the invention is centralized and does not require the use of main storage. This enables processors to communicate with each other in a fast and efficient manner. The arbitration circuitry prevents simultaneous access of the interprocessor communications facility by more than one processor, and decodes the commands sent from the processors and routes them to the processor interrupt circuitry or to the mailbox circuitry, depending on the command. The mailbox circuitry of the invention receives messages from sending processors and provides them to the intended receiving processors in a safe and secure manner. The processor interrupt circuitry facilitates the interprocessor communications process by handling interpro



EP 0 376 003 A3